XXX

XXX

Eindhoven University of Technology
xxx@tue.nl

Abstract. Keywords:

1 Literature Review

1.1 2012

IRIT Lab [4] defined two modules, one for context processing and the other for preference processing. The context processor uses Google Places API to filter by geographic position. They also defined 6 sets of categories and related it to different contexts. Every context consisted on season, time and day. After this context filtering they represented user positive and negative preferences using the Vector Space Model and then compare these vectors with the resulting places of the previous step using cosine similarity.

TNO and RUN [9] used Google Places API to collect “touristic attractions” in specific locations. For every user they built a positive and negative profile by extracting terms from the places they have rated using again Google Places API. Then they compared the vectors using cosine similarity. On a second run they used Kullback-Leibler divergence to rank the importance of the positive terms. Finally they re-ranked the results using Google Search to determine the category and popularity of a place.

In the University of Amsterdam [5] they downloaded and indexed documents from wikitravel. Then they took the positive descriptions from the user profile and used them to query their index.

In the University of Delawere [10] they extracted information of places from Yelp and Foursquare and then compute the similarity between user profiles and candidate places using category using formula (1) and descriptions using F2-EXP[3].

$$SIM_{\mathcal{C}}(e, C) = \frac{\sum_{c_i \in \mathcal{C}(e)} \sum_{c_j \in \mathcal{C}(C)} \frac{|Intersection(c_i, c_j)|}{max(|c_i|, |c_j|)}}{|\mathcal{C}(e)| \times |\mathcal{C}(C)|} \tag{1}$$

CSIRO [6] used Yahoo! Placefinder API to establish bounds in some cities and then collected places within some specific categories from Google Places API and Foursquare. Then the candidates are disambiguated. User profiles were represented as a bag of words weighted using BM25. Time sensitivity was associated using Foursquare check-ins, this way categories popular at some moment of the day or some specific days were found. Then both the user-based and time-based suggestions were mixed.

1.2 2013

The approaches are presented in form: from best result to worst.

In the paper [11], authors evaluated the effectiveness of (1) an opinion-based method to model user profiles and rank candidate suggestions; and (2) a template-based summarization method that leverages the information from multiple resources to generate the description of a candidate suggestion. The authors crawled the information such as candidate suggestions, main page, categories and reviews of candidates exclusively from Yelp for all contexts. They obtained the best result.

The paper [7] gets the second place. This system uses the Google Places API to obtain an initial list of suggestions. These were grouped into 27 different types. Description snippets were generated using the Yandex Rich Content API and Google Custom Search APIs. For each user a positive and negative model is generated, these models were based on the example descriptions which the system then expanded. These models were then used to rank suggestions lists for each profile-context pair. Authors used Naive Bayes as learning algorithm. Each suggestion in the training set is represented by a feature vector, consisting of two different types of features: boolean and real-valued. The boolean features were derived based on attraction types, representing the user's preferences with regard to the kind of venue suggested. Each place can be assigned to multiple types and as such our binary feature vector encodes the types each suggestion has been assigned to: 1 if it is assigned to that type, 0 otherwise. The 3 real-valued features were based on the cosine distance between the suggestion description and both user profiles (positive and negative), reflecting user term preferences, and the description length.

Both papers [11, 7] used the positive and negative models, we can estimate what the user's opinion might be about a potential suggestion based on its description.

The authors of [2] developed the system collects attractions from three commercial search engines, Google Places, Foursquare, and Yelp. The attractions returned by these services are enhanced by adding snippets from the Google and Bing search engines using crowdsourcing techniques. The first run submits each candidate place as a query in an index of examples and scores it based on the top-k users' preferences. The second run is based on Rocchio's algorithm and uses the examples per profile to generate a personal query which is then submitted to the index of attractions.

The runs udel_run_D, udel_run_SD [1] starts by analysing user profiles and generating bags of keywords depicting the user interests. Then, given a context, they query the Google Places API with each bag, retrieving lists containing places in that context which fit in the specified genre. Their first run creates suggestions for each profile-context combination by iterating through the lists in a round-robin fashion, selecting one place at a time to create a list of 50. Their second run sorts the retrieved lists by average user ratings before iterating round-robin.

The system [8] groups attractions into categories and determines which attraction categories users prefer by using the attraction ratings given in the user profiles. Suggestions are fetched using the Google Places API, passing the context location as a parameter. These suggestions are then ranked based on the distance between the suggestion and the context location and the level of preference the user has for the suggestion category. Suggestion descriptions are result snippets returned from Google when the suggestion named is passed as a query.

The university of Amsterdam team [1] developed the system that extracts suggestions for sightseeing, shopping, eating, and drinking from Wikitravel pages dedicated to US cities. Descriptions from positive examples in the user profiles are used as queries to rank suggestions. The system then merges the per-query rankings of positive examples into a single result list. These ranked suggestions are then filtered based on the context.

The 'IRIT.OpenWeb' [**?**] run ranks suggestions based on how close it is to the context and by considering the categories matched between the user and the suggestions. The system assigns users one or more categories from WordNet and Google Places based on which suggestions users liked. Attractions are retrieved for the 50 contexts from Google Places and also tagged with the same categories. Descriptions for suggestions were fetched using Yahoo! BOSS Placefinder and Bing. The 'IRIT.ClueWeb' [**?**] run is made up of documents retrieved using Terrier with queries composed of the users' categories. Documents are then ranked according to their retrieval score and similarity to the profile. Users are assigned to categories as in the 'IRIR.OpenWeb run and descriptions are also generated in the same way.

The system [1] CIRG finds candidate places from Google Places and WikiTravel. For each candidate place a corresponding description is extracted from the Google Places API or the Bing API. Related Wikipedia articles are also found for both example suggestions and candidate places based on the place's description. The system calculates an intersection of these Wikipedia articles between example suggestions and candidate places, these Wikipedia articles are then used to extract Wikipedia categories. A score based on the similarity of categories between candidate places and examples suggestions is calculated by the system, places with the highest score are then returned.

The system described in [**?**] gathers a candidate set of venues from the Yelp API. In the first run, uncsils_base, for each context-profile pair, the candidate venues are scored using the weighted average rating associated with the venues in the profile. For this calculation, each profile venue was given a weight based on the cosine similarity between the candidate venue and profile venue. The goal with this approach is to score each candidate venue based on the rating associated with the most similar venues in the profile. The second run, uncsils_param, boosted the contribution from the profile venue with the greatest similarity with the candidate venue and rating.

The authors of [**?**] This system uses Location-based Social Networks (LSNs), such as FourSquare and Facebook Places, to gather data on both venues and users. First, the similarity between the venue descriptions and a textual representation of the user's interests is calculated. Venue descriptions are extracted from their web pages or their profiles on LSNs. In the uogTrCFP run, on top of this similarity, the system focuses on using the social aspect in the ranking by incorporating an estimation of the popularity of the venue based on the number of previous interactions of the users on LSNs. The second run, uogTrCFX, also uses this popularity but focuses on using a personalisation model based on the XQuAD diversity framework, which allows the system to cover multiple categories of interest to the user.

The system [?] his system made used of data from Yelp for creating candidate suggestion and supplementing user pro

files. The system used vector space models to compute the similarity between candidates and examples and linear regression models to combine multiple attributes of candidate profiles into the calculations. The system was trained and tested using 5-fold cross validation on 2012 track data.

2 Useful Links

Google Places API - https://developers.google.com/places/documentation/

Yandex Rich Content API - http://api.yandex.com/rca/

Toolkit version 1.0 used, available from http://nltk.org/

Baseline (LambdaMART) - http://code.google.com/p/jforests/

3 Learnt Lessons from Results of 2013

Both methods [11, 7] (the best two results):

1. used the positive and negative models, we can estimate what the user's opinion might be about a potential suggestion based on its description;
2. used some natural language processing technique;
3. used open web (Yelp and Google places respectively);

Our plan:

1. We will use Fourthquare and (or) Facebook data. We need (1) reviews (2) user opinion (likes) Initially we want to join these two data sources but w are not sure that we can do that. Alejandro will work on this problem from 30.06 - 5.07.
2. Julia will use obtained datasets to construct users' profiles and construct features for a ranker (7.07 - 13.07)
3. Julia will come up with ranking function (the simplest on is presented in [11]) (07.07 - 13.07)
4. Learn from our mistakes and improve the results (15.07-28.07).

References

1. A. Bellogin, G. G. Gebremeskel, J. He, A. Said, T. Samar, A. P. de Vries, J. Lin, and J. B. P. Vuurens. Cwi and tu delft at trec 2013: Contextual suggestion, federated web search, kba, and web track. In *Proceedings of Twenty-Second Text REtrieval Conference (TREC 2013)*, 2013.
2. G. Drosatos, G. Stamatelatos, A. Arampatzis, and P. S. Efraimidis. Pdf file duth at trec 2013 contextual suggestion track. In *Proceedings of Twenty-Second Text REtrieval Conference (TREC 2013)*, 2013.
3. H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 480–487. ACM, 2005.
4. G. Hubert and G. Cabanac. IRIT at TREC 2012 Contextual Suggestion Track. Technical report, DTIC Document, 2012.
5. M. Koolen, J. Kamps, and H. Huurdeman. Contextual suggestion from wikitravel: Exploiting community-based suggestions. Technical report, DTIC Document, 2012.
6. D. Milne, P. Thomas, and C. Paris. Finding, weighting and describing venues: CSIRO at the 2012 TREC Contextual Suggestion Track. Technical report, DTIC Document, 2012.
7. A. Rikitianskii, M. Harvey, and F. Crestani. University of lugano at the trec 2013 contextual suggestion track. In *Proceedings of Twenty-Second Text REtrieval Conference (TREC 2013)*, 2013.
8. D. Roy, A. Bandyopadhyay, and M. Mitra. A simple context dependent suggestion system. In *Proceedings of Twenty-Second Text REtrieval Conference (TREC 2013)*, 2013.
9. M. Sappelli, S. Verberne, and W. Kraaij. TNO and RUN at the TREC 2012 Contextual Suggestion Track: Recommending personalized touristic sights using Google Places. Technical report, DTIC Document, 2012.
10. P. Yang and H. Fang. An exploration of ranking-based strategy for contextual suggestion. Technical report, DTIC Document, 2012.
11. P. Yang and H. Fang. An opinion-aware approach to contextual suggestion. In *Proceeding of of the Text REtrieval Conferences*, 2013.